CONWIP ASSEMBLY SYSTEMS WITH DETERMINISTIC PROCESSING AND RANDOM OUTAGES

IZAK DUENYAS
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

WALLACE J. HOPP
Department of Industrial Engineering and Management Sciences
Northwestern Unversity
Evanston, IL 60208

Technical Report 91-35

November 1991

CONWIP ASSEMBLY SYSTEMS WITH DETERMINISTIC PROCESSING AND RANDOM OUTAGES

IZAK DUENYAS
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109

WALLACE J. HOPP
Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston Illinois 60208

Abstract

We develop structural results and an approximation for the throughput of an assembly system fed by multi-station fabrication lines where releases are governed by the CONWIP protocol and all machines have deterministic processing times but are subject to random outages. This formulation is motivated by a printed circuit board manufacturing process.

We demonstrate that while throughput of such systems is nondecreasing in machine speed, there are cases where throughput declines when mean time between failures (MTBF) increases or mean time to repair (MTTR) decreases. Using the concept of "deterministic steady state," which describes the behavior of the system in the absence of failures, we derive a simple, closed-form aproximation for throughput. Comparisons with simulation show that this approximation is robust over a wide range of conditions. Finally, we observe that, throughput tends to be higher when the bottleneck is located in fabrication rather than assembly.

1 Introduction

Queueing network models are attractive tools for planning, design, and control of manufacturing systems because of their ability to capture and evaluate the effects of variability and congestion. Such models provide a useful complement to simulation (see Suri and Diehl 1987) and a basis for optimization. Because of their attractiveness, a number of commercial software packages have appeared and continue to appear on the market (see Snowdon and Ammons 1988 for a survey of queueing network packages).

One very real and important aspect of electronics and other manufacturing systems that has not yet been satisfactorily addressed in queueing network models is assembly operations. In the instance that motivated this study, multi-plane printed circuit boards (PCB's) are manufactured by fabricating the layers separately and then laminating them together. Despite the central role of assembly in this and other manufacturing systems, relatively little

analytical work has been done to model assembly-like queues. The bulk of the queueing models that explicitly handle assemblies represent the fabrication lines feeding the assembly operation as single machines, an assumption that limits their applicability to most manufacturing systems (Ammar 1980, Bhat 1986, Bonomi 1987, Hopp and Simon 1989, Lipper and Sengupta 1987, Simon and Hopp 1988). To date, the only analytic model of assembly systems that allows multiple station tandem fabrication lines feeding assembly is that by Duenyas and Hopp (1991). However, this model is restricted to the case where all processing times are exponential. The study of more general realistic systems has largely been limited to simulation studies (Baker, Powell and Pyke 1990a, 1990b). Because simulation by itself is tedious to use in a planning context and nearly impossible to use in a repetitive control context, there is a compelling need for better analytical models.

In queueing network models of tandem production, the issue of scheduling releases is often ignored (e.g., arrivals are assumed to occur according to a Poisson or other renewal process). However, if the fabrication lines feeding assembly are treated as independent arrival streams, an assembly-like queue can easily become unstable (Harrison 1977). Hence, it is important to explicitly represent the scheduling linkage between releases to the various fabrication lines feeding assembly. In many manufacturing systems, the method for controlling releases is MRP, in which releases are scheduled by subtracting fixed lead times from due dates. More recently, in the wake of the success of Japanese just-in-time methods, pull systems, such as kanban (Monden 1983, Ohno 1988), have become more widely used in industry.

Assembly systems operating under kanban can be modeled using Markov methods (see Hopp and Simon 1989). However, representing multi-station kanban fabrication lines feeding assembly requires a very complex Markov model, due to the blocking caused by kanban and the matching required at assembly. Instead, in this paper, we will restrict attention to the CONWIP (CONstant Work-In-Process) production control scheme, which offers many of the benefits of kanban (e.g., WIP control and more predictable outputs) but is simpler to model (see Spearman, Woodruff and Hopp 1989 and Spearman and Zazanis 1989 for discussions of CONWIP). In the simplest implementation of CONWIP, a new job is not started in a line until an existing job exits the line and jobs are pushed between machines in the line in first-come-first-served sequence. In an assembly system operating under CONWIP (illustrated in Figure 1), jobs exit only from assembly. Hence, all fabrication lines begin a new job whenever an assembly is completed. However, these simultaneously released jobs need not be intended for the same assembly, since it is quite possible that the fabrication lines have different WIP levels.

A second important modeling assumption required to represent an assembly system as a queueing network is the distribution of processing times on machines. One possibility, which is attractive from a computational standpoint, is the exponential. Duenyas and Hopp (1991) developed computational procedures and structural results for the assembly system operating under CONWIP where all fabrication and assembly times are exponentially distributed. But in many applications, including printed circuit board (PCB) manufacture, operations are either automated or highly structured, so that under normal conditions, processing times are nearly deterministic. Examples of PCB manufacturing processes that exhibit such regularity include lamination, trimming, drilling, copper plating, photographic expose, etching, coating, and optical inspection The variability in these operations comes more from outages than from any intrinsic variability in the process. To capture this type of

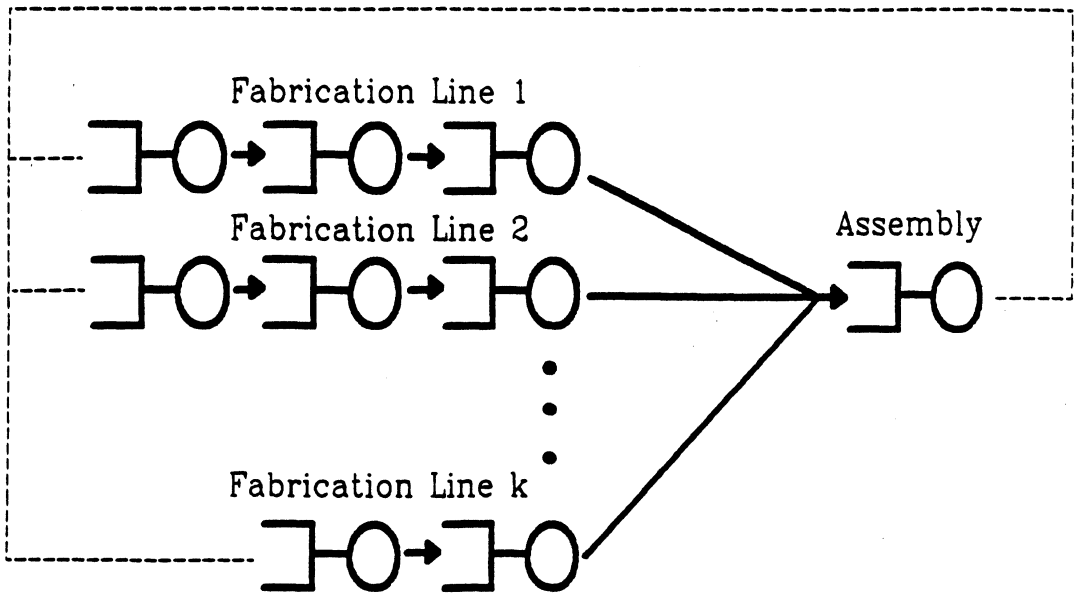

Figure 1: CONWIP Assembly System

behavior, we will assume that fabrication and assembly times are deterministic except when the machine is "down," and assume that times between failures and times to repair are randomly distributed. We term this assumption the deterministic-processing-random-outages (DPRO) assumption and note that it has been used in the modeling literature in a variety of contexts (see e.g., Akella and Kumar 1986, Buzacott 1971, Kimemia and Gershwin 1983, Sharifnia 1988, Wijngaard 1979).

Given that an assembly system operating under CONWIP and DPRO is of practical interest, it remains to arrive at a useful method for computing the parameters describing its behavior (in particular, throughput), so that we can derive planning and optimization procedures. One approach would be to replace the DPRO processing times with exponential processing times with matching means and use the previous work of Duenyas and Hopp (1991). However, as Hopp and Spearman (1991) demonstrate, replacing DPRO systems by corresponding exponential systems does not work well even for tandem closed queueing networks not involving assembly. There is no reason to expect such a substitution to work any better when assembly is considered. Furthermore, the approach of Duenyas and Hopp for exponential systems is an approximation, so there could be a problem of error compounding from the multiple levels of approximation. Because of this, we choose to analyze the DPRO system directly and derive an approximation based on the particular behavior of this system. The result is an analytically tractable means for computing throughput of CONWIP assembly systems, which represents the DPRO counterpart to the exponential results of Duenyas and Hopp.

The remainder of this paper is organized as follows. Section 2 introduces our notation and problem formulation. In Section 3, we derive some key structural results for the system, and in particular develop an approximate upper bound for throughput in an assembly

system. We use this upper bound to derive an approximation for the throughput in Section 4. In Section 5, we test our approximation against simulation for several different test problems. Section 6 concludes the paper.

2 Problem Formulation

We consider k fabrication lines feeding an assembly machine, where line j consists of m_j single-machine workstations and contains n_j jobs, as shown in Figure 1. We denote the machines in line j by (j,i), where $(j,1)$ is the first machine and (j,m_j) is the last machine in the line. Jobs are processed in order on machines (j,i), $i = 1,\ldots,m_j$ and then proceed to the assembly machine.

Under the DPRO assumption, service times are deterministic and we denote the service time of fabrication machine (j,i) as $\tau_{j,i}$. The service time of the assembly machine is also deterministic and is denoted by τ_A. Also under DPRO, machines are subject to failure and the time between failures for fabrication machine (j,i) is assumed exponential with rate $\lambda_{j,i}$. Repair times for fabrication machine (j,i) are also assumed exponential with rate $\mu_{j,i}$. Failures and repairs of the assembly are exponential with rates λ_A and μ_A, respectively.

The time a job spends at a machine (not in the queue) is taken to be the sum of its required processing time plus the times of any failures that occur during processing (i.e., failures induce a *preempt/resume,* rather than *preempt/repeat,* disruption of jobs). An output occurs when service at the assembly is completed. Under the CONWIP protocol, this sends a signal to machines $(j,1), j = 1,\ldots,k$, to add a new job to their queues.

We begin observing the output process at time $t = 0$ and define N_t as the number of outputs until time t. We are interested in finding the throughput for the system, $\theta = \lim_{t\to\infty} N_t/t$. By Little's law, the problem is equivalent to finding the average cycle time (flow time, round-trip time) for any one of the lines.

3 Structural Results

Let $\{\tau,\lambda,\mu/\tau_A,\lambda_A,\mu_A/n\}$ denote the assembly system shown in Figure 1 and let $\theta\{\tau,\ \lambda,\ \mu/\tau_A,\ \lambda_A,\ \mu_A/n\}$ represent the throughput of this system. We define τ to be a two-dimensional array containing the service times of all fabrication machines. Similarly, λ, and μ are arrays containing the failure and repair rates of the fabrication machines and n is an array containing the number of jobs in each line. For purposes of analysis, we define $\{\tau_r,\lambda_r,\mu_r/\tau_A,\lambda_A,\mu_A/n_r\}$ to be a closed tandem queueing network that consists of machines $(r,1),\ldots,(r,m_r)$ in sequence with the assembly machine at the end (where the assembly machine does not "assemble", but instead processes single jobs. In the hypothetical line corresponding to this network, jobs flow in sequence from machine $(r,1)$ to (r,m_r) and then to the assembly machine. To represent CONWIP, we assume that after completing work at the assembly, jobs return to machine $(r,1)$. Let $\theta\{\tau_r,\lambda_r,\mu_r/\tau_A,\lambda_A,\mu_A/n_r\}$ denote the throughput of this hypothetical system.

Our first structural result shows that the throughput of these hypothetical lines represent upper bounds on the throughput of the assembly system.

Proposition 1 $\theta\{\tau,\lambda,\mu/\tau_A,\lambda_A,\mu_A/n\} \le \min_r \theta\{\tau_r,\lambda_r,\mu_r/\tau_A,\lambda_A,\mu_A/n_r\}$

Proof: We begin by generating the successive failure and repair times at each machine (j,i) and at the assembly. We let $T^l_{j,i}$ denote the l_{th} downtime for machine (j,i). Similarly, we let T^l_A denote the l^{th} downtime for the assembly machine. We note that $T^l_{j,i}$ is a time interval. We let $T_{j,i}$ denote the union of these intervals for machine i in line j. We define $f_{j,i}(a, T_{j,i})$ as the time a job that starts being processed on machine i in line j at time a loses to machine failures. We let the successive service completion times for machines and assembly in system $\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$ be denoted by

$$X_{(j,i),1}, X_{(j,i),2}, X_{(j,i),3}, \ldots\ldots\ldots$$
$$X_{A,1}, X_{A,2}, X_{A,3}, \ldots\ldots\ldots$$

and those for system $\{\tau_r, \lambda_r, \mu_r/\tau_A, \lambda_A, \mu_A/n_r\}$ be denoted by

$$\hat{X}_{(r,i),1}, \hat{X}_{(r,i),2}, \hat{X}_{(r,i),3}, \ldots\ldots\ldots$$
$$\hat{X}_{A,1}, \hat{X}_{A,2}, \hat{X}_{A,3}, \ldots\ldots\ldots$$

Without loss of generality, we can start both systems with all jobs in front of the first machine in each line (machines $(j,1)$). Then for system $\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$, we have

$$X_{(j,i),p} = Y_{(j,i),p} + \tau_{(j,i)} + f(Y_{(j,i),p}, T_{j,i}) \tag{3.1}$$

where $Y_{(j,i),p}$ is the time at which the p^{th} job is ready to be processed at machine i of line j and is given by

$$Y_{(j,1),1} = 0 \tag{3.2}$$

$$Y_{(j,1),p} = X_{(j,1),p-1} \quad \text{for } 1 \le p \le n_j \tag{3.3}$$

$$Y_{(j,1),p} = max(X_{A,p-n_j}, X_{(j,1),p-1}) \quad \text{for } p > n_j \tag{3.4}$$

$$Y_{(j,i),p} = max(X_{(j,i-1),p}, X_{(j,i),p-1}) \quad \text{for } i = 2, \ldots, m_j \tag{3.5}$$

$$Y_{A,p} = max(\max_j X_{(j,m_j),p}, X_{A,p-1}) \tag{3.6}$$

Similarly for system $\{\tau_r, \lambda_r, \mu_r/\tau_A, \lambda_A, \mu_A/n_r\}$, we have

$$\hat{X}_{(r,i),p} = \hat{Y}_{(r,i),p} + \tau_{(r,i)} + f(\hat{Y}_{(r,i),p}, T_{r,i}) \tag{3.7}$$

where

$$\hat{Y}_{(r,1),1} = 0 \tag{3.8}$$

$$\hat{Y}_{(r,1),p} = \hat{X}_{(r,1),p-1} \quad \text{for } 1 \le p \le n_j \tag{3.9}$$

$$\hat{Y}_{(r,1),p} = \max(\hat{X}_{A,p-n_j}, \hat{X}_{(r,1),p-1}) \quad \text{for } p > n_j \tag{3.10}$$

$$\hat{Y}_{(r,i),p} = max(\hat{X}_{(r,i-1),p}, \hat{X}_{(r,i),p-1}) \quad \text{for } i = 2, \ldots, m_j \tag{3.11}$$

$$\hat{Y}_{A,p} = max(\hat{X}_{(r,m_r),p}, \hat{X}_{A,p-1}) \tag{3.12}$$

Suppose that for some positive integer p, we have

$$X_{(r,i),p} \ge \hat{X}_{(r,i),p} \quad \text{for } i = 1, \ldots, n_r \tag{3.13}$$

and

$$X_{A,p} \ge \hat{X}_{A,p} \tag{3.14}$$

Clearly, (3.13) and (3.14) hold for $p = 1$. Notice that for any a_1 and a_2 such that $a_1 \leq a_2$, $a_1 + \tau_{j,i} + f(a_1, T_{j,i}) \leq a_2 + \tau_{j,i} + f(a_2, T_{j,i})$. That is, if a job is available at any machine earlier to be processed, then it will come out earlier or at the same time as before, regardless of the machine's failure schedule. Using this fact along with (3.1), (3.4), (3.5), (3.6), (3.7), (3.10), (3.11), and (3.12), we get

$$X_{(r,i),p+1} \geq \hat{X}_{(r,i),p+1} \tag{3.15}$$

and

$$X_{A,p+1} \geq \hat{X}_{A,p+1} \tag{3.16}$$

Thus, we have shown that for all $p \geq 1$, we have $X_{A,p} \geq \hat{X}_{A,p}$. Since $p/X_{A,p}$ converges to $\theta\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$, and $p/\hat{X}_{A,p}$ converges to $\theta\{\tau_r, \lambda_r, \mu_r/\tau_A, \lambda_A, \mu_A/n_r\}$, the proposition is proven. □

Proposition 1 provides an upper bound for the throughput of the assembly system. Unfortunately, there is currently no computationally tractable way to calculate the throughput of a tandem closed queueing network under the DPRO assumption exactly. However, Hopp and Spearman (1991) have developed an approximation for such a system. We can use their approximation along with Proposition 1 to obtain an approximate upper bound for the assembly system.

Let $\tau_{(r,b)}$ represent the processing time of the slowest machine in line r disregarding failures (i.e., $b = argmax_i \tau_{(r,i)}$). We can use the Hopp and Spearman approximation by treating the assembly machine as the last machine for each line, just as we did in the proof of Proposition 1. That is, we consider k different tandem closed queueing networks, where network i ($i = 1, \ldots, k$) consists of $n_i + 1$ machines where the $n_i + 1^{st}$ machine is the assembly machine. (Again, the assembly machine does not "assemble," but acts like an additional fabrication machine at the end of each line). Then their approximation for $\theta\{\tau_r, \lambda_r, \mu_r/\tau_A, \lambda_A, \mu_A/n_r\}$, which we denote by $\hat{\theta}_r$ is given by

$$\hat{\theta}_r = \begin{cases} \frac{\sigma_r}{\tau_{rb}} & \text{if } m_r > M_r \\ \varepsilon_r \frac{m_r}{\sum_{i=1}^{n_r+1} \tau_{r,i}} & \text{otherwise} \end{cases} \tag{3.17}$$

where

$$M_r = \frac{\sum_{i=1}^{n_r+1} \tau_{r,i}}{\tau_{(r,b)}}$$

$$\sigma_r = \frac{1 + \sum_{i \neq b} \lambda_{r,i}(\frac{1}{\mu_{r,i}} - \beta_{r,i})}{1 + \sum_{i \neq b} \frac{\lambda_{r,i}}{\mu_{r,i}}}$$

$$\beta_{r,i} = \frac{exp(-\mu_{r,i} z_r)}{\mu_{r,i} \tau_{(r,b)}} [\tau_{r,i} + \frac{1}{\mu_{r,i}}(1 - exp[-\mu_{ri}(\tau_{(r,b)} - \tau_{r,i})])]$$

$$z_r = m_r \tau_{(r,b)} - \sum_{i=1}^{n_r+1} \tau_{r,i}$$

$$\varepsilon_r = \frac{1 + \sum_{i \neq b} \lambda_{r,i}(\frac{1}{\mu_{r,i}} - \gamma_{r,i})}{1 + \sum_{i \neq b} \frac{\lambda_{r,i}}{\mu_{r,i}}}$$

$$\gamma_{r,i} = \frac{1}{m_r \tau_{(r,b)} + g_r \mu_{r,i}} \frac{1}{}[m_r(\tau_{r,i} + 1/\mu_{r,i}) - \frac{exp[-\mu_{r,i}(\tau_{(r,b)} - \tau_{r,i})]}{\mu_{r,i}}(m_r - 1 + exp(-\mu_{r,i} g_r))]$$

$$g_r = -z_r$$

Hence, by Proposition 1, an approximate upper bound on $\theta\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$ is

$$\hat{U} = \min_r \hat{\theta}_r.$$

In the next section, we use this bound to develop an approximation for the throughput of the assembly system.

In the same manner in which we proved that the hypothetical tandem lines, $\{\tau_r, \lambda_r, \mu_r/\tau_A, \lambda_A, \mu_A/n_r\}$, $r = 1, \ldots, m_r$, achieve higher throughputs than the assembly system, we can prove the (rather obvious) result that throughput is a nonincreasing function of processing times.

Proposition 2 *If* $\hat{\tau} \geq \tau$ *and* $\hat{\tau_A} \geq \tau_A$, *then* $\theta\{\hat{\tau}, \lambda, \mu/\hat{\tau_A}, \lambda_A, \mu_A/n\} \leq \theta\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$

Proof: The proof is very similar to that of Proposition 1 and is omitted. □

One is tempted to use the same sample path proof approach to demonstrate that throughput increases whenever failure rates, λ and λ_A decrease or repair rates, μ and μ_A increase. However, this apparently intuitive result is not even true for all DPRO systems.

As a simple counter-example, consider a two machine tandem network with one job and suppose that both machines have deterministic processing times equal to one minute. Further suppose that only machine 1 experiences failures and they last exactly one minute and occur with a spacing of one minute (i.e., from end of one failure to beginning of the next). If we start the system with the job at machine 1 and the first failure occurs at $t = 1$, then failures occur only when the job is being processed at machine 2, and have no effect on throughput. Hence, a job will exit every two minutes, for a throughput of 0.5 jobs/minute (30 jobs/hour).

Now suppose we reduce the failure times (i.e., the duration of the repair) from one minute to 30 seconds in duration. Then the first failure occurs at $t = 1$, the second at $t = 2.5$, the third at $t = 4$, and so on. But, in the previous case, a job was processed on machine 1 during the interval between $t = 2$ and $t = 3$. Hence, the failure at $t = 2.5$ causes this job to be delayed. Similar delays occur in subsequent jobs. In fact, it is simple to show that every other job is delayed at machine 1 for 30 seconds. Hence, a job is output on average every 2.25 minutes, for a throughput of 0.44 jobs/minute (26.67 jobs/hour). Shorter failures cause throughput to decrease in this case.

This same example can be used to show that longer times between failures can cause throughput to decrease. For instance, if failures remain of duration 1 minute, but times between failures are increased to 1.5 minutes, the result is to delay each job at machine 1 (other than the first job) by 30 seconds. Hence, the output rate falls to 1 job every 2.5 minutes for a thoughput of 0.4 jobs/minute (24 jobs/hour).

This counter-intuitive behavior is also exhibited by assembly systems with more than one fabrication line and more than two stations per line if the failure and repair times are deterministic. Presumably there are some conditions on the independence and randomness of the failure times that would ensure that throughput is increasing in repair rate and decreasing in failure rate. However, it would seem that the sample path proof technique used for Propositions 1 and 2 is not adequate for demonstrating this result.

4 An Approximation for Throughput

In this section, we derive an approximation for $\theta\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\}$. To do this, we make two levels of approximations. First, we observe that

$$\theta\{\tau, \lambda, \mu/\tau_A, \lambda_A, \mu_A/n\} \simeq \frac{\mu_b}{\mu_b + \lambda_b}\hat{\theta} \tag{4.18}$$

where $\hat{\theta}$ is the throughput of the system with when the bottleneck machine (which is defined by $b = argmax_{(j,i)}\tau_{(j,i)}$) is completely reliable. However, calculating $\hat{\theta}$ directly is not easy, so we make a second level of approximation.

Our second level of approximation is based on the idea of "deterministic steady state" (DSS) introduced by Hopp and Spearman (1991). A deterministic assembly system in which all lines have at least a minimum number of jobs reaches a cyclic behavior, which we term DSS, in which the bottleneck machine never becomes starved and the amount of work at each machine moves through the same cycle over and over again. The minimum number of jobs required in line j to ensure that the bottleneck never starves is M_j, where $M_j = (\tau_A + \sum_{i=1}^{n_j}\tau_{(j,i)})/\tau_b$. If $m_j \geq M_j, j = 1, \ldots, k$, then the bottleneck machine never becomes starved in (DSS), however all non-bottleneck machines will be cyclically starved, and the eventual cycle of work at each machine is independent of the initial distribution of WIP. If $m_j < M_j$, for some $j = 1, \ldots, k$ then a DSS cycle will be reached, but the bottleneck will starve and the nature of the cycle may depend on the initial distribution of WIP. (See Hopp and Spearman (1991) for further discussion of DSS.)

One difference between the tandem closed queueing network considered by Hopp and Spearman (1991), and the assembly system here is that there are critical WIP levels for each line M_j (in a tandem closed queueing network, there is only one critical WIP level, since there is only one line). Furthermore, these critical WIP levels depend on the location of the bottleneck. In particular, it makes a difference whether assembly, which is shared by all lines, or one of the fabrication machines is the bottleneck. Because of this, we develop two different approximations for $\hat{\theta}$ under these two different situations. By using the resulting approximations in (4.18), we get an approximation for the throughput of the assembly system.

4.1 Bottleneck at Assembly

We first consider the case where the assembly machine is the bottleneck, i.e., $\tau_A > \tau_{j,i}, j = 1, \ldots, k, i = 1, \ldots, n_j$. In this case, $M_j = (\tau_A + \sum_{i=1}^{n_j}\tau_{ji})/\tau_A$. What this means is that if each line j has m_j jobs, where $m_j > M_j$, then assembly is never starved under DSS. Therefore, as long as there are no failures, the system's throughput will be $1/\tau_A$, since the bottleneck will always be busy. Because, the bottleneck emits one job every τ_A time units, it acts to pace the line, so that all the other machines will get a new job every τ_A time units, will take $\tau_{j,i}$ time units to process it, and hence will be busy $\tau_{j,i}/\tau_A$ of the time.

WIP Above M_j

We can use the above properties of (DSS) to derive an approximation for the case where the assembly machine is the bottleneck and $m_j > M_j$ for all lines, as follows:

$$\hat{\theta} \approx \frac{\delta}{\tau_A} \tag{4.19}$$

where δ is the fraction of the time for which the assembly is not starved in the system where the bottleneck is completely reliable, and the following assumptions hold:

(A1) Each time a non-bottleneck machine fails, no other non-bottleneck machine fails until that machine is repaired.

(A2) At the time of a non-bottleneck machine failure, WIP in the system is equally likely to be in any configuration along the DSS cycle.

Notice that (4.19) would be exact if δ represented the true fraction of time assembly is not starved. Unfortunately, δ cannot be computed in any straightforward fashion, so we approximate it by performing our computations under the conditions of Assumptions (A1) and (A2). Hence, the quality of our overall approximation depends on the extent to which these assumptions are naturally satisfied. In general, if failure rates are low with respect to repair rates, and the rate at which DSS is reached, then Assumptions (A1) and (A2) will aproximate reality. In the next section, we present simulation results under different conditions to provide a direct test of their utility.

To use Assumptions (A1) and (A2) to derive an expression for δ, we note that under Assumption (A1) the system alternates between a period where all the machines are up, and one where one non-bottleneck machine is down. By virtue of Assumption (A2), the point where the failed machine has been repaired and all the machines are "up" is a regeneration point. Then, invoking renewal reward theory, $\delta = E[P_1]/E[T_1]$, where T_1 is the time of the first regeneration and $E[P_1]$ is the expected time work is available at the bottleneck during a regenerative cycle.

We can compute the expected length of the regenerative cycle as the sum of the expected down time (which will depend on which machine failed) plus the expected up time of all machines, which yields,

$$E[T_1] = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \frac{\lambda_{j,i}}{\Lambda} \frac{1}{\mu_{j,i}} + \frac{1}{\Lambda} \tag{4.20}$$

where

$$\Lambda = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \lambda_{j,i}$$

To calculate the expected time that work is available at the bottleneck during a cycle, we first note that a job from line j, arriving at assembly will spend $z_j = (m_j - 1)\tau_A - \sum_{i=1}^{n_j} \tau_{j,i}$ waiting to begin assembly. This time represents a cushion against WIP gaps caused by failures of machines in that line. That is, as long as that particular job is not delayed for more than z_j time units, the assembly machine will not be starved. Under Assumptions (A1) and (A2), there is no interaction between failures, and DSS is reached after each failure. Hence, each time a failure occurs in one of the machines, the cushion represented by z_j is the only factor that protects the assembly machine.

Of course, in reality, sometimes failures occur before DSS is reached. For example, a failure in one line may starve the assembly causing jobs from other lines to wait in front of the assembly machine. If a failure occurs in one of the other lines at that point, then the WIP cushion in that line may very well be greater than z_j. We examine the extent to which such events "average out" with respect to throughput in our simulation tests in Section 5.

To calculate δ, under Assumption (A1) and (A2), we define X to be a random variable representing the length of the outage at the assembly machine caused by the failure of non-bottleneck machine (j,i). For each station (j,i), we define a cyclic clock where time 0 corresponds to the start of the busy period and ends at the end of the idle period. We let U be the time of the failure on this cyclic clock. Thus, the failure will either occur in a busy (B) period, or an idle (I) period. If a failure occurs while the machine is idle, then we let V represent the length of time until the next job arrives to the machine. We denote the time the machine is down by Y. Then, as in Hopp and Spearman (1991), we write

$$E[X|U,V,Y] = \begin{cases} max\{0, Y - z_j\} & \text{if } U \in B \\ max\{0, Y - z_j - V\} & \text{if } U \in I \end{cases} \tag{4.21}$$

We uncondition on Y by noting that Y is exponentially distributed with rate $\mu_{j,i}$, and on U by noting that $P(U \in B) = \tau_{j,i}/\tau_A$, $P(U \in I) = (\tau_A - \tau_{j,i})/\tau_A$ and that $\{U \in I\}$ implies that V is uniformly distributed on $[0, \tau_A - \tau_{j,i}]$. Then, defining $\beta_{j,i}$ to be the expected value of X given that machine (j,i) failed,we get

$$\beta_{j,i} = \frac{exp(-\mu_{j,i} z_j)}{\mu_{j,i}\tau_A}[\tau_{j,i} + \frac{1}{\mu_{j,i}}(1 - exp[-\mu_{j,i}(\tau_A - \tau_{j,i})])] \tag{4.22}$$

Hence, the expected time work is available at the bottleneck during a regenerative cycle is given by:

$$E[P_1] = \sum_{j=1}^{k}\sum_{i=1}^{n_j} \frac{\lambda_{j,i}}{\Lambda}(\frac{1}{\mu_{j,i}} - \beta_{j,i}) + \frac{1}{\Lambda} \tag{4.23}$$

Using (4.23) and (4.20), we can write δ, the fraction of time that the bottleneck is not starved by nonbottleneck failures as,

$$\delta = \frac{1 + \sum_{j=1}^{k}\sum_{i=1}^{n_j} \lambda_{j,i}(\frac{1}{\mu_{j,i}} - \beta_{j,i})}{1 + \sum_{j=1}^{k}\sum_{i=1}^{n_j} \frac{\lambda_{j,i}}{\mu_{j,i}}} \tag{4.24}$$

By using (4.24), along with (4.18), and (4.19), we get an approximation for the throughput of the assembly system, θ_{a1}, where

$$\theta_{a1} = \frac{\mu_A}{\mu_A + \lambda_A}\frac{\delta}{\tau_A} \tag{4.25}$$

Finally, we note that we can refine our approximation by incorporating the approximate upper bound $\hat{U}$, derived in the previous section. In some cases, θ_{a1} may be greater than U. However, $\hat{U}$ is an approximate upper bound. Hence, it is reasonble to use θ_{ap} as an approximation for the assembly system, where

$$\theta_{ap} = min\{\theta_{a1}, \hat{U}\} \tag{4.26}$$

WIP Below M_j

If one or more of the lines have WIP levels that are lower than the critical WIP levels for those particular lines, then the bottleneck will periodically starve in DSS. The system will be driven by the slowest line. To approximate the throughput in this case, we first consider all

the lines that have WIP levels below M_j. Suppose line j has WIP level $m_j < M_j$. Then, if we considered the closed queueing network that consists of line j and the assembly machine, the throughput of this closed queueing network, with no failures, would be $m_j/(\sum_{i=1}^{n_j} \tau_{j,i} + \tau_A)$. The DSS cycle in this case would depend on the initial distribution of WIP. If all the jobs in this regular closed queueing network with no failures were at the assembly when we start the process, then the DSS cycle will involve exactly one starvation period of length $g = \sum_{i=1}^{n_j} \tau_{j,i} - (m_j - 1)\tau_A$ of the bottleneck per round trip of job. If we start with other initial WIP allocations, this starvation period may be broken into several intervals. We ignore this latter case, since it only complicates the analysis without affecting the accuracy of our approximation.

In the assembly system with no failures, the throughput will be governed by the slowest line. That is, for all lines that have $m_j < M_j$, we calculate $m_j/(\sum_{i=1}^{n_j} \tau_{j,i} + \tau_A)$, and the throughput of this system with no failures will be the minimum of these values. The WIP in the slowest line will exhibit the behavior described above for a tandem closed queueing network. (Since this is the slowest line and there are no failures, a job from this line arriving at assembly will always find a job from the other lines.) That is, if line r is the slowest line, and we start the system with all the jobs in each line in front of assembly, then the assembly machine will have a cycle where it processes m_r jobs, then it will be idle for $g = \sum_{i=1}^{n_r} \tau_{r,i} - (m_r - 1)\tau_A$ time units. Then the same cycle will begin again. For all other machines in line r, there will be a period, of duration, $\tau_{r,i}$, where the machine is processing a job and then an idle period of duration $\tau_A - \tau_{r,i}$. Every m_r jobs, the idle period will be of duration $\tau_A - \tau_{r,i} + g$. Hence, a failure in one of the machines in line r will starve assembly if it occurs while a job is being processed, or if it occurs while the machine is idle *and* it lasts more than $\tau_A - \tau_{r,i} + g$ or $\tau_A - \tau_{r,i}$ time units depending on the duration of that particular idle period.

The effects of a failure of a non-bottleneck machine in any other line than r on the bottleneck machine is slightly more complex. A job from any other line always arrives to the assembly machine earlier than a job from line r in the system with no failures. On average, a job from line $j, j \neq r$ will wait $z_j = m_j \frac{\sum_{i=1}^{n_r} \tau_{r,i} + \tau_A}{m_r} - \tau_A - \sum_{i=1}^{n_j} \tau_{j,i}$ time units after it joins the assembly queue before its assembly begins. We note that this is an average value. However, in the following approximation, we will treat it as if it is exact (that is, as if a job from line j will wait z_j time units every time.) In general, WIP levels below M_j are not likely to be used in practical situations, and as we will show later, our simple approximation for this case works well.

Having defined z_j as above (with $z_r = 0$ for the slowest line) we define, for each machine in the network, I to represent the idle times with duration $\tau_A - \tau_{j,i}$ and $\hat{I}$ to represent the idle time with duration $\tau_A - \tau_{j,i} + g$. Defining X, Y, V and U as in the previous section, we get

$$E[X|U,V,Y] = \begin{cases} max\{0, Y - z_j\} & \text{if } U \in B \\ max\{0, Y - V - z_j\} & \text{if } U \in I \\ max\{0, Y - V - z_j\} & \text{if } U \in \hat{I}. \end{cases} \tag{4.27}$$

We uncondition on Y, U and V by noting that $P(U \in B) = m_r\tau_{j,i}/(m_r\tau_A + g)$, $P(U \in I) = (m_r - 1)(\tau_A - \tau_{j,i})/(m_r\tau_A + g)$, $P(U \in \hat{I}) = (\tau_A - \tau_{j,i} + g)/(m_r\tau_A + g)$. We define as

previously, $\beta_{j,i}$ to be the expected value of X given that machine (j,i) failed. Then, we get

$$\beta_{j,i} = \frac{exp(-\mu_{j,i}z_j)}{m_r\tau_A + g}\frac{1}{\mu_{j,i}}[m_r(\tau_{j,i}+1/\mu_{j,i}) - \frac{exp[-\mu_{j,i}(\tau_A - \tau_{j,i})]}{\mu_{j,i}}(m_r - 1 + exp(-\mu_{j,i}g))] \quad (4.28)$$

We can use (4.28) in (4.24) to get δ, the fraction of time that the bottleneck is not starved by the failure of a non-bottleneck machine. Then, our approximation for the throughput is

$$\theta_{a1} = \frac{\mu_A}{\lambda_A + \mu_A}\delta\frac{m_r}{\sum_{i=1}^{n_r}\tau_{r,i} + \tau_A} \quad (4.29)$$

Finally, we again refine our approximation by using $\hat{U}$ and (4.26).

4.2 Bottleneck in Fabrication

We next consider the case where the bottleneck is in one of the fabrication lines. The analysis in this case is similiar to the previous case. Assume that the bottleneck is in line r, and its processing time is τ_{rb}. We first derive the critical WIP levels. For line j, this is given by $M_j = (\tau_A + \sum_{i=1}^{n_j}\tau_{j,i})/\tau_{rb}$.

If $m_j > M_j$ for all j, then the bottleneck does not starve in DSS. Our analysis is similiar to the previous section. A job in line r, will spend $z_r = m_r\tau_{br} - \sum_{i=1}^{n_r}\tau_{ri}$ time units in the queue in front of the bottleneck. Hence, a job in line r can be delayed z_r time units by the failure of a non-bottleneck machine in line r, or by the failure of the assembly machine, without causing the bottleneck to starve. If a machine in any of the other lines fails, then this will delay the arrival of that job to the assembly machine. But, as long as a job is not delayed in the assembly machine by more than z_r time units, this does not starve the bottleneck.

The one critical difference between this case and that with the bottleneck at assembly is that we must also consider the fact that the jobs in the faster lines spend some time in front of the assembly machine waiting for jobs from line r. This wait is equal to $m_j\tau_{rb} - \tau_A - \sum_{i=1}^{n_j}\tau_{ji}$ time units. Hence, the amount of time that a job in line j can be delayed without causing starvation in the bottleneck machine in line r is equal to $z_j = z_r + m_j\tau_{rb} - \tau_A - \sum_{i=1}^{n_j}\tau_{ji}$.

Having defined the z_j values in this manner, the rest of the analysis is the same as in the previous subsection, and the approximation is given by

$$\theta_{ap} = max\{\theta_{a1}, \hat{U}\}$$

where

$$\theta_{a1} = \frac{\mu_{rb}}{\mu_{rb} + \lambda_{rb}}\frac{\delta}{\tau_{rb}} \quad (4.30)$$

where

$$\delta = \frac{1 + \sum_{(j,i)\neq(r,b)}\lambda_{j,i}(\frac{1}{\mu_{j,i}} - \beta_{j,i}) + \lambda_A(\frac{1}{\mu_A} - \beta_A)}{1 + \sum_{j=1}^{k}\sum_{i=1}^{n_j}\frac{\lambda_{j,i}}{\mu_{j,i}} + \frac{\lambda_A}{\mu_A}}$$

and

$$\beta_{j,i} = \frac{exp(-\mu_{j,i}z_j)}{\mu_{j,i}\tau_{rb}}[\tau_{j,i} + \frac{1}{\mu_{j,i}}(1 - exp[-\mu_{j,i}(\tau_{rb} - \tau_{j,i})])]$$

The approximation for the case where $m_j < M_j$ for one or more lines in this case is similiar to that of the previous case. Since, as we noted before, M_j is typically a very low WIP level from a practical standpoint, since it causes the bottleneck to starve regularly even without failures, we refrain from discussing this case in more detail.

5 Computational Results

The real test of any approximation is how well it works over a range of cases. To test the approximation described above, we generated a variety of problems, and compared the throughput of the system from simulation, θ_s, with our approximation results. To provide a balanced test, we tried to consider scenarios that are both "good" and "bad" with regard to satisfying Assumptions (A1) and (A2). Assumption (A1) will be approximately satisfied if machine availabilities are high. Assumption (A2) will tend to hold if times between failures are long relative to processing times and the bottleneck is sharp (so that WIP accumulates in its DSS configuration rapidly). Hence, we would expect our approximation to work well for unbalanced systems with high machine availabilities. The critical question we must address with our simulation analyses is how robust the approximation is for systems that do not have these characteristics.

The cases summarized in Table 1 represent a range of scenarios that might be encountered in practice. Case 1 represents a system with perfectly balanced fabrication and assembly, high availability (90.91%), and assembly and relatively long, infrequent failures. (In real circuit board manufacturing applications, availabilities are typically even higher (e.g., in excess of 95%) and failures are typically longer and less frequent, which would tend to make our approximation work even better than indicated by our "high availability" examples.) Case 2 represents a system with a pronounced bottleneck at assembly, high availabilities (88.89%), and shorter, more frequent failures. Case 3 is identical to Case 1 except that the bottleneck has lower availability (66.7 %). Case 4 is identical to Case 2, except that the bottleneck has been exchanged between assembly and fabrication. Case 5 represents a "large" example, with two 5–machine fabrication lines feeding assembly, high availabilities with long, infrequent failures, and a less pronounced bottleneck at assembly. Case 6 represents a less reliable system, with availability of assembly of only 55.5% and availability of fabrication machines of only 71.43%, with short frequent failures. This last example is deliberately intended to cause the approximation to work poorly. Case 7 is identical to Case 5 except that the bottleneck has been exchanged between assembly and fabrication.

For each case, we examined the accuracy of the approximation for a variety of WIP allocations. We made use of a MOR-DS (Curry, Deuermeyer, Feldman 1989) program to simulate the systems for different WIP allocations. For each case, we simulated the system for 100000 time units 10 times and recorded the throughput each time. The throughput we report represents the average of the 10 runs. Each simulation run (10 values) lasted about 8 minutes on a 486 machine, while the computation involved in our approximation took negligable time (less than 1 second for each of the examples considered). The values resulting from simulation from our approximation are reported in Tables 2 through 6.

Case 1 is well-suited to the approximation by virtue of having high availability with long infrequent failures, but is poorly suited to it by virtue of being balanced. Evidently, the favorable failure and repair characteristics offset the balance problem to a large extent. As shown in Table 2, the approximation works well for this example, with the maximum error of 4.3%.

Example 2 is in some sense the reverse of Example 1. This example is well-suited to the approximation by virtue of having a sharp bottleneck at assembly, but is poorly suited to it by virtue of having short frequent failures. Interestingly, as shown in Table 3 the

Parameter	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
τ_A	1	2	1	1	3.5	2	0.8
$1/\mu_A$	100	8	20	8	150	10	150
$1/\lambda_A$	10	1	10	1	10	8	10
τ_{11}	1	1	1	2	1.7	0.5	1.7
$1/\lambda_{11}$	100	8	100	8	150	10	150
$1/\mu_{11}$	10	1	10	1	10	4	10
τ_{12}	1	1	1	1	2.1	1	2.1
$1/\lambda_{12}$	100	8	100	8	150	10	150
$1/\mu_{12}$	10	1	10	1	10	4	10
τ_{13}	-	1	-	1	1.3	-	1.3
$1/\lambda_{13}$	-	8	-	8	150	-	150
$1/\mu_{13}$	-	1	-	1	10	-	10
τ_{14}	-	-	-	-	0.6	-	0.6
$1/\lambda_{14}$	-	-	-	-	150	-	150
$1/\mu_{14}$	-	-	-	-	10	-	10
τ_{15}	-	-	-	-	2.5	-	2.5
$1/\lambda_{15}$	-	-	-	-	150	-	150
$1/\mu_{15}$	-	-	-	-	10	-	10
τ_{21}	1	1	1	1	0.8	0.8	3.5
$1/\lambda_{21}$	100	8	100	8	150	10	150
$1/\mu_{21}$	10	1	10	1	10	4	10
τ_{22}	1	1	1	1	1.4	1	1.4
$1/\lambda_{22}$	100	8	100	8	150	10	150
$1/\mu_{22}$	10	1	10	1	10	4	10
τ_{23}	-	1	-	1	3.0	-	3.0
$1/\lambda_{23}$	-	8	-	8	150	-	150
$1/\mu_{23}$	-	1	-	1	10	-	10
τ_{24}	-	-	-	-	1.0	-	1.0
$1/\lambda_{24}$	-	-	-	-	150	-	150
$1/\mu_{24}$	-	-	-	-	10	-	10
τ_{25}	-	-	-	-	1.1	-	1.1
$1/\lambda_{25}$	-	-	-	-	150	-	150
$1/\mu_{25}$	-	-	-	-	10	-	10

Table 1: Summary of Data for Examples.

n_1	n_2	θ_s	θ_{ap}	%err
3	4	0.643	0.661	2.8
3	3	0.634	0.649	2.4
2	2	0.432	0.436	0.9
4	4	0.647	0.674	4.2
5	5	0.667	0.696	4.3
2	3	0.440	0.447	1.6

Table 2: Results for Example 1.

n_1	n_2	θ_s	θ_{ap}	%err
2	2	0.284	0.280	-1.5
3	3	0.383	0.387	1.0
4	3	0.400	0.412	3.0
4	4	0.428	0.437	2.1
5	3	0.405	0.415	2.5
5	4	0.434	0.440	1.4
5	5	0.442	0.443	2.3

Table 3: Results for Example 2.

approximation also works well for this case, with a maximum error around 3%. It seems that having either a distinct bottleneck or high availabilities with long infrequent failures is sufficient for the approximation to perform well.

Clearly, another way in which the bottleneck can be sharp, all other things being equal, is for it to have a lower availability than the other machines. If this is the case, then, as in the case where the bottleneck has longer processing times, WIP will tend to accumulate quickly at the bottleneck and hence the system will frequently be in (or close to) DSS at the time of nonbottleneck failures. In Example 3, we tested this assumption. The fabrication machines in this example are identical to those in Example 1. The assembly machine has the same processing time as in Example 1 but it fails more often, resulting in an availability of 66.7%. Evidently, a sharp bottleneck due to availability also causes the approximation to work well, as evidenced by Table 4, which shows a maximum error of 1.6%.

In Example 4, we consider the same system as Example 2, but with the bottleneck machine in one of the fabrication lines. Specifically, we exchange the assembly machine with machine 1 in fabrication line 1. As shown in Table 5, the approximation has similar accuracy to that exhibited for Example 2. Apparently, the location of the bottleneck does not critically affect the accuracy of the approximation. Note, however, that the throughput *is* affected by the location of the bottleneck. The throughput for the case with the bottleneck in fabrication is higher than that for the case with the bottleneck in assembly for every allocation of WIP. We discuss this issue further below.

In Example 5 the fabrication lines are not balanced and the system is larger than all of the previous examples, consisting of a total of 11 machines. Moreover, the bottleneck,

n_1	n_2	θ_s	θ_{ap}	%err
2	2	0.323	0.320	-1.0
2	3	0.333	0.328	-1.6
3	3	0.468	0.476	1.7
3	4	0.482	0.485	0.6
4	4	0.498	0.494	-0.8
5	5	0.505	0.511	1.2

Table 4: Results for Example 3

n_1	n_2	θ_s	θ_{ap}	%err
3	3	0.400	0.411	2.8
4	3	0.429	0.440	2.6
3	4	0.404	0.415	2.7
4	4	0.434	0.440	1.4
4	5	0.437	0.441	0.9
5	4	0.441	0.444	0.7
5	5	0.443	0.444	0.2

Table 5: Results for Example 4.

which is at assembly is not as pronounced as in Examples 2 and 4. In spite of these factors, however, the approximation works well, with errors for all WIP allocations within 3%, as shown in Table 6.

Examples 1 through 5 had all machines with availabilities close to 0.90. Even though in practice, availabilities are typically this high or higher, we tested our approximation against simulation for a system with low availabilities to see how much it is degraded. Example 6, as we noted earlier, has availability of assembly of only 55.5% and availability of fabrication machines of only 71.43% with short frequent failures. However, it does have a sharp bottleneck at assembly. Table 6 shows that while the approximation does not work

n_1	n_2	θ_s	θ_{ap}	%err
2	2	0.109	0.109	0.0
3	3	0.156	0.155	-0.7
3	4	0.404	0.415	2.7
4	4	0.434	0.440	1.4
4	5	0.437	0.441	0.9
5	4	0.441	0.444	0.7
5	5	0.443	0.444	0.2

Table 6: Results for Example 5.

n_1	n_2	θ_s	θ_{ap}	%err
3	2	0.167	0.161	-3.6
3	3	0.193	0.190	-1.6
3	4	0.202	0.208	3.0
4	3	0.200	0.206	3.0
4	4	0.218	0.224	2.8
5	5	0.236	0.246	4.3
3	5	0.210	0.219	4.3

Table 7: Results for Example 6.

as well as in the previous examples, which is expected, since DSS would virtually never be reached in this system, the approximation still yields results within 4.3% of the simulation values.

Finally, we return the the issue that arose in our discussion of Examples 2 and 4 regarding the placement of the bottleneck. There we observed that if we interchange the assembly machine with a fabrication machine thereby moving the bottleneck to fabrication the throughput went up. This result, if general, would have implications for allocation of resources in a fabrication/assembly operation. For instance, if a manager has the option of moving a worker from fabrication to assembly, which would decrease capacity of fabrication but increase it at assembly, should he/she do it?

We can address this question via our approximation and through simulation. First, to use the approximation, we begin by considering the case with no failures. Clearly, under these conditions, if $m_r > M_r$ for each line, then the position of the bottleneck does not matter since the system will produce at the rate of the bottleneck regardless of the position of the bottleneck. However, when there are failures, our approximate analysis indicates that it is better to have the bottleneck operation in one of the fabrication lines.

As an illustration of why this is the case, consider a system with two fabrication lines. Let z_1, and z_2 be the time cushions against failures for line 1 and line 2 for this system. Suppose we switch the assembly machine with machine i in line 1. Denote the new time cushions as z_1' and z_2'. It is clear that $z_1' = z_1$ and $z_2' = z_2 + \tau_A - \tau_{1i} + z_1 \geq z_2$. Since the time cushion against failures for line 2 increases, we would expect the throughput for this system to be higher.

Next, to see whether the behavior predicted by the approximation for this simple case holds for more general examples, we ran a number of simulations. In each case, we found that the throughput increased when the bottleneck was moved from assembly to fabrication. Examples 2 and 4 illustrate one such example. As another, compare Example 7 with Example 5. The only difference between Examples 7 and 5 is that the bottleneck has been moved from assembly to the first machine of fabrication line 2. Table 8 compares the simulated throughputs for the two systems with θ_1 denoting the throughput of the system in Example 5 and θ_2 denoting the throughput of the system in Example 7. These results again show that moving the bottleneck from assembly to fabrication increase the throughput for every WIP allocation. In this particular example, the increase in throughput was as high as 21.3%.

n_1	n_2	θ_1	θ_2	%dif
2	2	0.109	0.118	8.3
3	3	0.156	0.169	8.3
2	4	0.122	0.148	21.3
3	4	0.166	0.185	11.1
4	4	0.185	0.197	6.5
5	4	0.195	0.204	4.6
5	5	0.203	0.217	6.9

Table 8: Effect of Location of Bottleneck.

6 Conclusions

We have given a simple, closed-form expression that relates the throughput of an assembly system with DPRO machines operating under the CONWIP protocol to the allocation of WIP in the various fabrication lines. This approximation works particularly well when there is a sharp bottleneck and times between failures are long relative to processing rates. However, it appears to be quite robust to violation of these conditions.

By using our approximation and the results leading up to it, we have also made the following structural observations concerning CONWIP assembly systems under the DPRO assumption:

1. Throughput is an increasing function of processing rates.
2. Throughput is not necessarily an increasing function of repair rates or failure times.
3. Given an identical set of machines, throughput tends to be higher if the bottleneck is located in fabrication rather than assembly.

7 References

Akella, R., P,R. Kumar. 1986. "Optimal Control of Production Rate in Failure Prone Manufacturing System." *IEEE Transactions on Automatic Control* **AC-31**, 116.

Ammar, M.H. 1980. "Modelling and Analysis of unreliable manufacturing assembly networks with finite storage." MIT Laboratory for Information and Decision Sciences, Report LIDS-TH-1004.

Baker, K.R., S.G. Powell and D.F. Pyke. 1990a. "Buffered and Unbuffered Assembly Systems with Variable Processing Times." Working Paper No. 246, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH 03755.

Baker, K.R., S.G. Powell and D.F. Pyke. 1990b. "Optimal Allocation of Work in Assembly Systems." Working Paper No. 258, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH 03755.

Bhat, U.N. 1986. "Finite capacity assembly-like queues." *Queueing Systems: Theory and Applications* **1** 15, 85.

Bonomi, F. 1987. "An approximate analysis for a class of assembly-like queues." *Queueing Systems: Theory and Applications* **1**, 289.

Buzacott, J.A. 1971. "The role of Inventory Banks in Flow-Line Production Systems," *International Journal of Production Research* **9**, 425.

Curry, G.L., B.L. Deuermeyer, R.M. Feldman. 1989. *Discrete Simulation: Fundamentals and Microcomputer Support.* Holden Day, Oakland, CA.

Duenyas, I., W.J. Hopp. 1991. "Estimating the Throughput of a Cyclic Exponential Assembly System," Technical Report 91-14, Dept. Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Duenyas, I., W.J. Hopp, M.L. Spearman. 1991. "WIP and Quota Setting in a Hierarchical Pull Production System." Technical Report 91-11, Dept. Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Harrison, J.M. 1973. "Assembly-like Queues." *Journal of Applied Probability* **10**, 354.

Hopp, W.J., J.T. Simon. 1989. "Bounds and Heuristics for Assembly-Like Queues." *Queueing Systems: Theory and Applications* **4**, 137.

Hopp, W.J., M.L. Spearman. 1991. "Throughput of a Constant Work in Process Manufacturing Line Subject to Failures." *International Journal of Production Research* **29**, 635.

Kimemia, J., and S.B. Gershwin. 1983. "An Algorithm for the Computer Control of a Flexible Manufacturing System." *IIE Transactions* **15**, 353.

Lipper E.H. , B. Sengupta. 1986. "Assembly-Like Queues with Finite Capacity: Bounds, Asymptotics and Approximations." *Queueing Systems: Theory and Applications* **1**, 67.

Monden, Y. 1983. *Toyota Production System.* Industrial Engineering and Management Press, Norcross, GA.

Ohno, T. 1988. *Toyota Production System: Beyond Large Scale Production.* Productivity Press, Cambridge, MA (original Japanese 1978).

Sharifnia, A. 1988. "Production Control of a Manufacturing System with Multiple Machine States." *IEEE Transactions on Automatic Control* **33**, 620.

Simon, J.T., W.J. Hopp. 1988. "Availability and Average Inventory in Balanced Assembly-Like Flow Systems." forthcoming in *IIE Transactions.*

Snowden, J.L., J.C. Ammons. 1988. "A Survey of Queueing Network Packages for the Analysis of Manufacturing Systems," *Manufacturing Review* **1**, 14.

Spearman, M.L., W.J. Hopp and D.L. Woodruff. 1989. "A Hierarchical Control Architecture for Constant Work-In-Process (CONWIP) Production Systems." *Journal of Manufacturing and Operations Management* **2**, 147.

Spearman, M.L., D.L. Woodruff and W.J. Hopp. 1990. "CONWIP: A Pull Alternative to Kanban." *International Journal of Production Research* **28**, 879.

Spearman, M.L., M.A. Zazanis. 1989. "Push and Pull Production Systems: Issues and Comparisons." to appear in *Operations Research.*

Suri, R., G.W. Diehl. 1987. "Rough Cut Modeling: An Alternative to Simulation." *CIM Review* **3**, 25.

Wijngaard, J. 1979. "The Effect of Inter-Stage Buffer Storage on the Output of Two Unreliable Production Units in Series with Different Production Rates." *AIIE Transactions* **11**, 42.